VLISP-80

C'est peut-être pas déconnant de faire un interprète LISP sur µprocess de type 8080 ou Z80.

On devrait pouvoir approcher des performances de l'actuel VLISP T1600, en temps & place, en utilisant le nouvel interprète SVLISP sans CØNS qui brime un peu les gens mais pas trop.

NOUVEAU

SVLISP

IL UTILISE LES NØUVEAUX LANCEMENTS SUPER RAPIDES

FAIS LA CHASSE AUX CONS

NE PAS ADMINISTER PAR VOIE BUCCALE

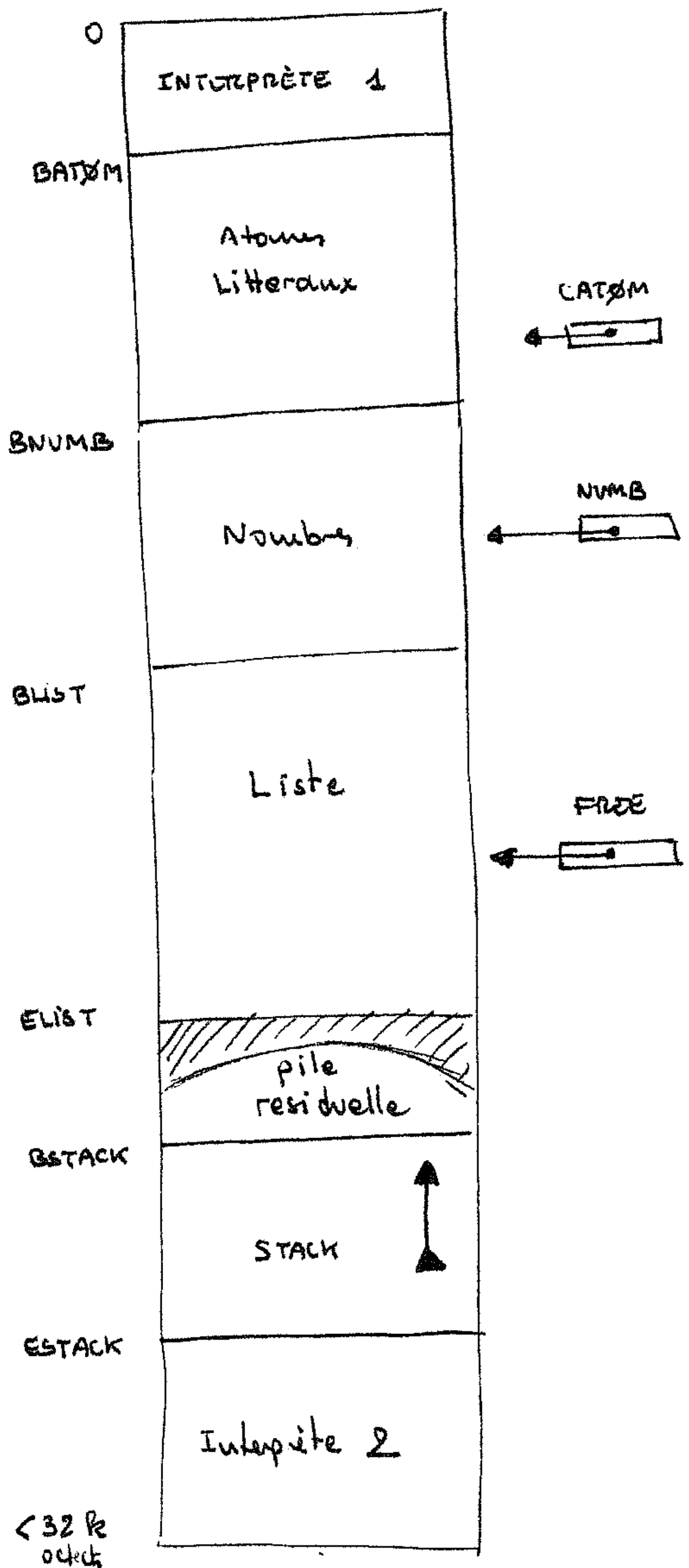
JC

Aout 77

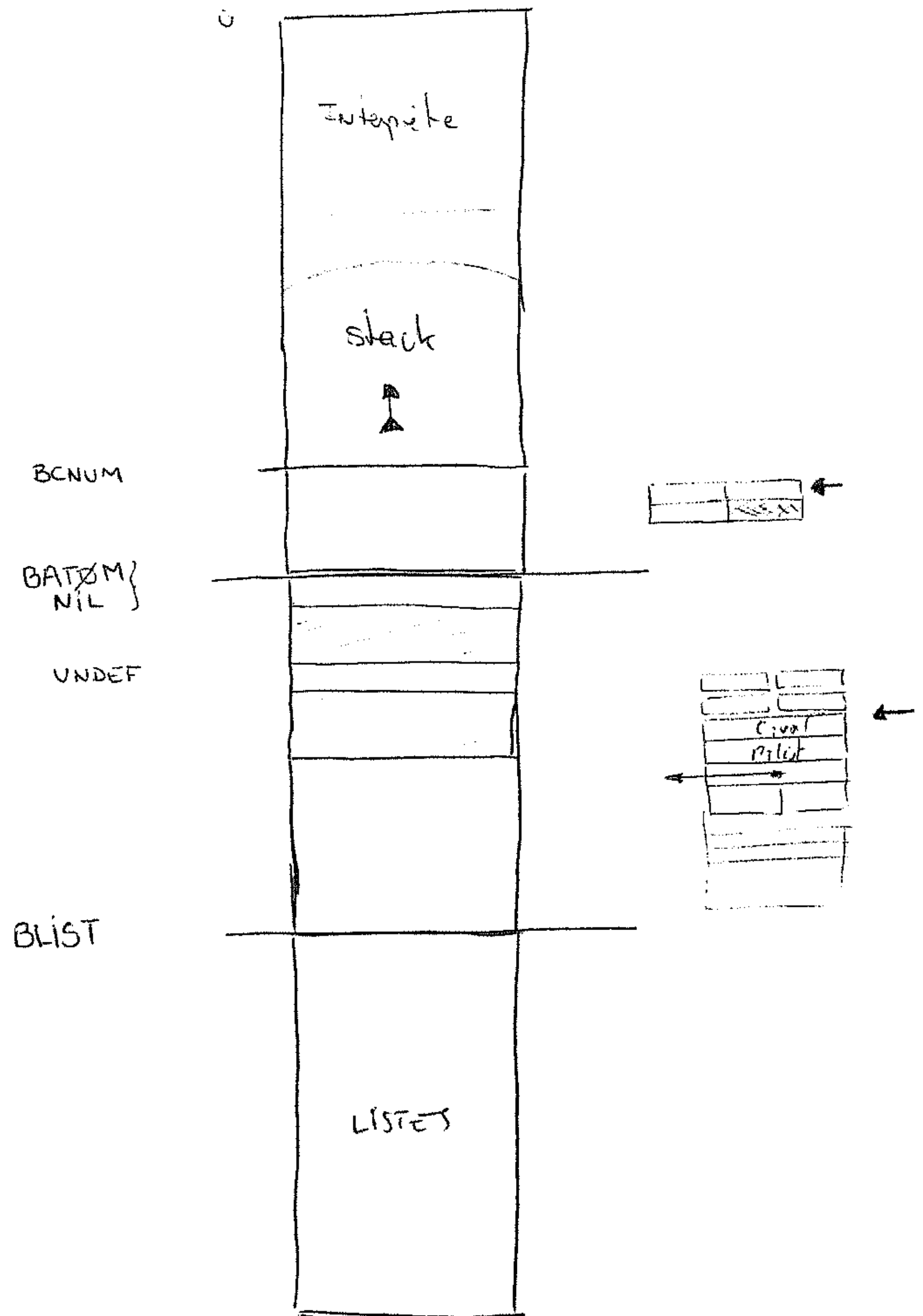
la memoire

l'Organisation de la memoire n'est pas très originale, mais ça a fait ses preuves.

Pour bien faire, il faudrait 32k octects.

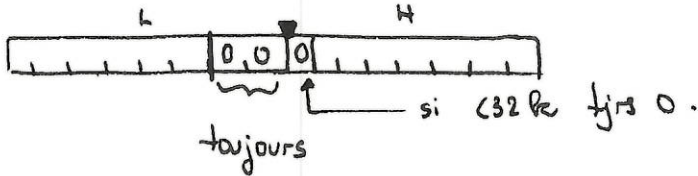


La memoire

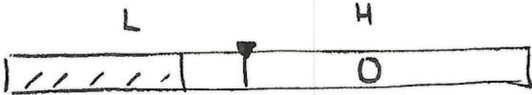


Les pointeurs

de type



NIL



Un test de type se fait toujours en comparant la partie H des pointeurs

Macros de test de type



ex: si DE = List vers LAB

```
[6]     5  MØV   A,D
        7  CPI   BLIST
[22]   10  JNC   LAB
```

si HL = NIL return

```
[3]     4  XRA   A      ; A ← 0
14/20   5  ØRA   H      ; si H=0 Z
     5/11  RZ
```

ex: si HL ≠ LIST RETURN

```
[4]      5  MØV   A,H
         7  CPI   BLIST
17/23 5/11  RC
```

Les Atomes littéraux

forme externe : classique (à la création C-VAL = UNDEF)
ou
"xxxx -- xxx" (à la création C-VAL = eux-mêmes)
C'est pas la peine de quoter les pseudos-chaînes

Pname < 64 χ

forme interne

Pointeur sur atome @ x 4.

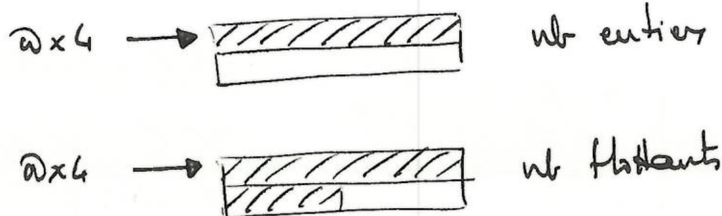
L , H	**F-VAL :** @ des SUBRs ou ((larg) body) des DEFs
	F-Typ : type codé de la fonction si y a : 0SUBR 1SUBR 2SUBR 3SUBR FSUBR λ 0λ 1λ Fλ Mλ
	P-Typ : bit pseudo-chaîne, nb de χ du P-name.
L , H	**C-VAL :** classique
L , H	**P-LIST :** classique
L , H	**A-Link :** lien vers l'atome suivant (permet le hash-code)
P-n 1 , P-n 2 / P-n 3 , P-n 4 / P-n 5 , P-n 6 / ... / Pn n-3 , P-n n-2 / P-n n-1 , P-n n	**P-name :**

Pas de G.C. des atomes littéraux

Les atomes numériques

forme externe : nb decimaux signés (par la suite, les flottants "à la Basic")

forme interne

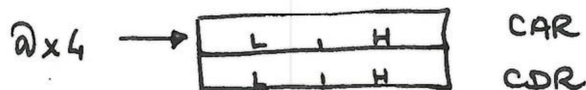


tous les nb sont alloués par liste libre dans la zone nb + G-C.

les listes

forme externe : classique avec paires pointées.

forme interne :

@x4 →


CAR
CDR

doublets alloués par liste libre & G.C.

manières d'utilisation des listes

[n] n octets
[n] (bleu) n cycles 8080
[n] (rouge) n cycles Z80

CAR (CAR HL) → BC

3 / 19 / 18

```
77 MØV  C,M
45 INR  L
77 MØV  B,M
```

(CAR HL) → HL

4 / 24 / 22

```
77 MØV  A,M
45 INR  L
77 MØV  H,M
45 MØV  L,A
```

CDR (CDR HL) → DE

5 / 29 / 26

```
45 INR  L
45 INR  L
77 MØV  E,M
45 INR  L
77 MØV  D,M
```

(CDR HL) → HL

6 / 34 / 30

```
45 INR  L
45 INR  L
77 MØV  A,M
45 INR  L
77 MØV  H,M
45 MØV  L,A
```

UNCØNS (CAR HL) → HL
& (CDR HL) → DE

9 / 52 / 48

```
77 MØV  E,M
45 INR  L
77 MØV  D,M
45 INR  L
77 MØV  A,M
45 INR  L
77 MØV  H,M
45 MØV  L,A
44 XCHG
```

RPLACA DE → (CAR HL)

3 / 19 / 18

```
77 MØV  M,E
45 INR  L
77 MØV  M,D
```

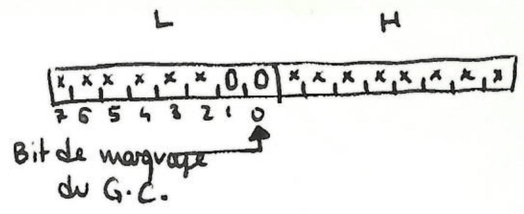
RPLACD DE → (CDR HL)

5 / 29 / 26

```
45 INR  L
45 INR  L
77 MØV  M,E
45 INR  L
77 MØV  M,D
```

le G.C.

tous les pointeurs sur des objets LISP ont la forme :



le G.C. utilise les algorithmes classiques de marquage & demarquage

marquage de l'objet LISP → HL

Z80 — 23 / 134

```
MARK1: MØV  E,M
     8 BIT  0,E
  5/11 RNZ
    15 SET  0,(HL)
     4 INR  L
     7 MØV  D,M
     4 INR  L
     7 MØV  A,M
     4 INR  L
     7 MØV  H,M
     4 MØV  A,L
     4 XCHG
    11 Push de
    17 CALL mark
    10 Pop  HL
MARK:
     8 BIT  7,H
    12 JR   NZ,MARK1

RET ou { RET
```

8080 — 25 / 138

```
MARK1: 7 MØV  A,M
       5 MØV  E,A     ; E ← (CAR HL)L
       4 RAR          ; déjà marqué ?
    5/11 RC           ; oui : c'est fini
       4 STC          ; cary ← 1
       4 RAL          ; positionne le bit G.C.
       7 MØV  M,A     ; range (CAR HL)L
       5 INR  L       ;
       7 MØV  D,M     ; DE ← (CAR HL)
       5 INR  L
       7 MØV  A,M
       5 INR  L
       7 MØV  H,M
       5 MØV  A,L
       4 XCHG         ; HL ← (CAR HL) ; DE ← (CDR HL)
      11 Push DE
      17 CALL mark    ; recurse sur les CARs
      10 POP  HL      ; itère sur les CDRs
MARK:  5 MØV  A,H     ; HL liste ?
       7 CPi  BLIST   ;
      10 JNC  MARK1   ; oui

RET ou { marquage special atome
       { RET
```

(remplace le bloc XCHG / Push DE) — 173

```
    11 Push HL      ; sauve (CDR)
    10 Lxi  hL,0    }
    10 Dad  SP      }
     5 mov  A,H     }  test fin de pile
     7 CPi  BSTAK   }
    10 JC   ERFS in G.C. }
     4 XCHG         ; HL ← (CAR)
```

EVAL

- est la seule f_n interprète (ya plus d'APPLY).
- ne fait aucun CONS (sauf dans le cas des LEXPRs où l'on demande explicitement EVLIS par la notation pointée).
- utilise les lancements super-rapides generalisées à tous les types de fonction.
- force: lancement

0SUBR	250 c	1c = 0.5 µs sur 8080	
1SUBR	270 c		parfois 0.25 µs sur Z80
FSUBR	250c		
Expr à 1 arg	790 c		

- traite les λ-expression de type EXPR FEXPR & MACROs
- est compatible avec les autres interprètes VLISP (en particulier pour la redefinition des fonctions ...)
mais ne permet plus les effets de bord consistant à faire evaluer des arguments supplementaires qui n'étaient pas utilisés.
De plus les definitions de fonctions ne sont plus sur les P-listes (mais dans la F-VAL). Pour {obtenir / ou manipuler} (le corps des fonctions il faut utiliser les f_n SETF & GETF.
- ya plus de NSUBRs (trop lourd)
par ex: LIST devient la forme FSUBRée de EVLIS .....
et quand y fo, ce sont les f_n elles mêmes qui demandent d'evaluer les arguments 1 à 1.
les NSUBRs sont devenus FSUBRs
(c'est plus commode et ya pas de CONS inutiles).

Organisation des fonction

toutes les fonctions rendent leur valeur dans H&L et rentrent par un RETurn.

à l'appel

1SUBR	l'argument est dans HL
2SUBR	1er arg → HL, 2ème arg → DE
3SUBR	1er arg dans la pile, 2ème arg → HL, 3ème arg → DE
FSUBR	l'argument est dans HL.

EVAL 1 SUBR

```
EVALCA1:  XCHG            ; (EVAL (CAR de))
[28]
EVALCA:   MØV   A,M       ; (EVAL (CAR hl))
[24]      INR   L         (CAR HL) → HL
          MØV   H,M
          MØV   L,A

EVAL:   5  MØV   A,H
22      7  CPI   BLIST    si HL est liste vers EVAL1
       10  JNC   EVAL1

        7  CPI   BNUM     cas nombre
     5/11  RNC

        7  MØV   E,M
        5  INR   L        (CAR HL) → DE
        7  MØV   D,M        C-VAL

        4  XCHG           ; CVAL → HL

        5  MØV   A,H
        7  CPI   ]'UNDEF
     5/11  RNZ
        5  MØV   A,L
        7  CPI   ['UNDEF
     5/11  RNZ
           DCR   E
           [ER A8] nom de l'atome → DE.

EVAL1:     ; cas forme = liste

        7  LXI   DE,0
        4  XCHG
       10  DADD  SP
47      5  MOV   A,H      test fin pile.
        7  CPI   BSTACK
       10  JC    ERFS
        4  XCHG
```

73 pour nb & NIL

```
 5  mov   A,H
 7  cpi   BATØM
5/11 RNC
 7  CPI   BLIST
10  JNC   eval 1
```

nb: 40

atome: 80

```
 7  mov   E,M
    ~~cpi   UNT~~
 5  INR   L
 7  MØV   D,M
 4  XCHG
 5  mov   A,H
 7  cpi   UNDEF
11  RNZ
```

46

80

```
16   SHLD    FØRM          ; sauve la forme pour les macros (& les erreurs)
EVAL11:
52   VNCØNS  HL HL DE      ; HL ← la fnt, DE ← les args.
5    MØV     A,H
7    CPI     BNUMB         ; si fnt ≠ ATØM → eval 2
10   JNC     EVAL2
5    DCX     HL            ; HL pointe sur P-Typ
5    DCR     L             ; HL pointe sur F-Typ
7    MØV     A,M           ; A ← F-Typ
5    DCR     L
7    MØV     B,M
5    DCR     L             ; récupère la F-VAL et l'empile.
7    MØV     C,M
11   PUSH    BC

7    MVI     H,]TEVAL
5    MØV     L,A           ; branchement indexé en f du F-TYP.
7    MØV     L,M             (DE contient le CDR de la forme).
7    MVI     H,]zone EVAL
4    PCHL
----
241
```

			Type		
TEVAL :	DB	ERAG	0		
en OD	251	PØPJ	1	OSUBR	
	eval + 269	EVALCA1	2	1SUBR	
	eval + eval + 408	EV2	3	2SUBR	
	eval + eval + eval + 535	EV3	4	3SUBR	
	251	PØPJ	5	FSUBR	
	748 (1 var)	Fλ	6	FLAMBDA	Fexpr
	eval + 781	Mλ	7	MLAMBDA	MACRØ
	470	0λ	8	0LAMBDA	EXPR à 0 arg
	eval + 789	1λ	9	1LAMBDA	EXPR à 1 arg
	eval + 666	Lλ	10	LLAMBDA	EXPR (type LEXPR)
		λ	11	LAMBDA	

```
EV3:          ; lancement 8SUBRs
      4  XCHG
     52  UNCONS   HL HL DE
EVARG 11 PUSH     DE            ; sauve le CDR
     17  CALL     EVAL          ; evalue le 1er arg
     10  POP      DE            ; remp le CDR
     18  XTHL              }
     11  PUSH     HL       }
      4  XCHG                   ; prepare le reste
EV2:          ; lancement 2SUBRs
     52  UNCONS   HL HL DE
EVARG 11 PUSH     DE
     17  CALL     EVAL
     18  XTHL
24   17  CALL     EVALCA
     10  POP      DE
      4  XCHG                   ; HL 1er arg , DE 2eme.
POPJ:         ; lancement 0SUBR & FSUBR
     10  RET
```



```
EVAL2:   CPI  BLIST     ; la fct est un at?
         JNC  EVAL3     ; non.
         XCHG           ; HL ← (larg)
         PUSH DE        ; sauve le at
         CALL EVCAR     ; evalue l'argument
         POP  DE
         XCHG
         MOV  C,M       ; C ← le no de l'elemt <128 actuellement.
         XCHG
         JMP  EVAL22
EVAL21:  INR  L
         INR  L
         MOV  A,M       ; (CDR HL) → HL
         INR  L
         MOV  H,M                                CNTH implicit.
         MOV  L,A
         MOV  A,M
         CPI  BLIST     ; fin liste?
         RC             ; ouaip.
EVAL22:  DCR  C         ; on compte
         JP   EVAL21    ; y fo encore avancer
         JMP  CAR

EVAL3:   MOV  A,M
         CPI  [λ                        la fct est donc une liste
         JNZ  EVAL31
         INR  L         ; c'est une λ explicite?
         CPI  ]λ
         JZ   ____      ; ouaip
EVAL31:  DCR  L         ; non: repositionne L
         PUSH DE
         CALL EVAL      ; on evalue la fct.
         POP  DE
         JMP  EVAL11    ; sa roule.
```

```
0λ:        HL ← ∞          F-VAL [illegible](body) empilé

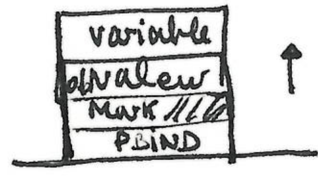
  16 LHLD    PBIND
  18 XTHL
  10 LXI     BC,-1
  11 PUSH    BC
  10 JMP     1λ5

     μλ:   LHLD    FØRM   ; l'argument des macros
        16 ~~JMP~~ F λ     est la forme elle-même.
        17 CALL
        10 JMP     eval

1λ:        DE ← (arg) non evalué,   F-VAL ((var) body) empilé

Fλ:  41 CALL    EVALCAL
~~[illegible]~~: 5 MØV     B,H        ; BC ← l'argument evalué ou prêt.
      5 MØV     C,L

     16 LHLD    PBIND
     18 XTHL
     10 LXI     DE,-1
     11 Push    DE
     52 UNCØNS  HL DE HL
     16 S~~T~~HLD   BØDY
1λ3: 4 XCHG
     52 UNCØNS  HL HL DE     ; 1er var → HL reste dans DE
     11 PUSH    HL           ; empile la variable
      7 MØV     A,M
      5 INR     L            ; HL ← old c-val
      7 MØV     H,M
      5 MØV     L,A
     18 XTHL                 ; oldval ↔ var
     11 push    HL           ; donc →
      7 MØV     M,C
      5 INR     L            ; rplaca var val
      7 MØV     M,B
        ~~JMP~~
        ~~UNCØNS  HL HL DE~~
        ~~PUSH    HL~~
      7 LXI     BC,0    }
        ~~XCHG~~        }
      5 MOV     A,H     }   liaison des var locales
      4 ØRA     L       }   si y en a.
     10 JNZ     1λ3     }
     16 LHLD    BODY
```



; les arguments sont liés, HL ← BODY

```
1λ5:    LXI     DE,O
        XCHG
        DADD    SP
        CPI     BSTACK
47      MOV     A,H
        CPI     BSTACK
        JC      ERFS
   16   SHLD    P.BIND
    4   XCHG
LESCAPE: 13 CALL PROGN
    4   XCHG
   10   JMP     1λ7
                                        UNBIND
1λ6: 10 POP     HL
    7   MOV     M,C
    5   INR     L
    7   MOV     M,B
1λ7: 10 POP     BC
    5   MOV     A,B
    5   INR     A
   10   JNZ     1λ6
   10   POP     HL
   16   SHLD    PBIND
    4   XCHG
   10   RET
193
                LESCAPE:  CALL   PROGN
                          XCHG
                          LHLD   PBIND      ;
   LESCAPE:               XRA    A
                          ORA    H          ; ya pas eu de Bind !!
                          JZ     ER LESCAPE
                          SPHL
                          XCHG
                          JMP    1λ7
```

Lλ: lexpr

```
     4  XCHG              ; HL ← les args
    17  call  EVLIS
     5  mov   B,H         ; arg evalué → BC
     5  mov   C,L
Lλ1: 16 LHLD  PBIND
    18  XTHL              ; prepare la pile
     7  LXi   DE,-1
    11  PUSH  DE
    52  UNCONS HL HL DE   ; HL ← var, DE corps
    11  PUSH  HL
     7  MOV   A,M
     5  INR   L           ; HL ← old c-val
     7  MOV   H,M
     5  MOV   L,A
    18  XTHL
    11  Push  HL          ; reempile le nom
     7  MOV   M,C
     5  INR   L           ; place
     7  MOV   M,B
     4  XCHG
    10  JMP   Lλ5
   ---
   232
```

```
         PUSH  HL
ESCAPE:  LXi   BC, '::ESC
         JMP   Lλ1
```

```
PRØGDX:            ;;; (EPRØGN (CDR DE))

        XCHG       ; HL ← DE
PRØGD:             ;;; (EPRØGN (CDR HL))
        INR  L
PRØGD1: INR  L
        MØV  E,M        ; (CDR HL) → DE
        INR  L
        MØV  D,M

PRØGNX:            ;;; (EPRØGN DE)

        XCHG       ; HL ← DE

PRØGN:             ;;; (EPRØGN HL)        tail recursif
        XRA             ; test NIL 1er fois
        ØRA  H
        JNZ  PRØGN2
        RET             ; NIL is NIL
PRØGN1:
        PUSH DE
        CALL EVAL
        PØP  HL
PRØGN2:
        MØV  E,M    \
        INR  L
        MØV  D,M
        INR  L          ; UNLØNS HL HL DE
        MØV  A,M
        INR  L
        MØV  H,M
        MØV  A,L
        XCHG        /

        XRA             ; si DE ≠ NIL vers PRØGN1
        ØRA  D
        JNZ  PRØGN1
        JMP  EVAL       ; pour la tail recursion.
```

EVPRED	IF	IFN
	FSUBR	FSUBR

```
EVPRED:         ; (EVAL (CAR HL))
                ; retour (CDR HL) → DE , Z si eval = NIL
                                         NZ si eval ≠ NIL

        MØV   E,M
        INR   L
        MØV   D,M
        INR   L
        MØV   A,M     ; UNLØNS HL HL DE
        INR   L
        MØV   H,M
        MØV   L,A
        XCHG
        PUSH  DE      ; sauve le CDR
        CALL  EVAL    ; evalue le CAR
        PØP   DE      ; restaure le CDR
        XRA           ; test le resultat de eval
        ØRA   H       ; par rapport à NIL
        RET           ; voilà.
```

```
IF:             ; (IF pred then else ....)      tail-recursif.

        CALL  EVPRED  ; evalue le predicat
        XCHG          ; HL ← (then else ...)
        JNZ   EVCAR   ; clause then
        JMP   PRØGD   ; clause else
```

```
IFN:            ; (IFN pred then else - - )     tail recursif

        CALL  EVPRED
        XCHG
        JZ    EVCAR   ; then
        JMP   PRØGD   ; else
```

CØND FSUBR   SELECTQ FSUBR

```
CØND1: UNCØNS HL HL DE    ; HL ← clause suivante
       PUSH   DE          ; sauve le reste des clauses
       CALL   EVPRED      ; evalue le test de la clause
       JNZ    CØND2       ; c'est la bonne.
       PØP    HL          ; recupère le reste des clauses
CØND:                     ; CØND complet   tail recursif
       XRA
       ØRA    H           ; fin des clauses ?
       JNZ    CØND1       ; non.
       RET                ; oui : old *ER A3.
CØND2: XRA                ; ya quekchose à faire ?
       ØRA    D
       JNZ    PRØGNX      ; oui.
       RET                ; non : ou ramène la val du test.

                          ; petit SELECTQ   tail recursif
SELECTQ: CALL EVPRED        ne marche qu'avec des litatoms.
       MØV    B,H
       MØV    C,L         ; BC ← la val du selecteur.
SELEC1: XCHG              ; HL ← les clauses
       UNCØNS HL HL DE
       XRA
       ØRA    D           ; c'est la dernière
       JZ     PRØGN       ; oui : clause failed.
       MØV    A,M         ; sinon compare le CAR de la clause
       CPR    C
       JNZ    SELEC1      ; c'est pas cui-là.
       INR    L
       MØV    A,M
       CPR    B
       JNZ    SELEC1      ; c'est pas cui-là.
       JMP    PRØGD1      ; c'est ØK.
```

non tail recursif

```
ØR:     CALL  EVPRED      ; evalue le terme suivant
        RNZ               ; c'est ≠ de NIL, je rentre
        XRA
        ØR
        XCHG              ; HL ← les termes suivants.
        XRA
        ØRA   H           ; si en reste ?
        JNZ   ØR          ; ouais.
        RET               ; non.
```

non tail recursif

```
AND1:   XCHG              ; HL ← les termes suivants.
AND:    CALL  EVPRED      ; evalue le terme suivant
        RZ                ; c'est = NIL je rentre
        XRA
        ØRA   D           ; si en reste ?
        JNZ   AND1        ; ouais.
        RET               ; non (HL ← la dernière evaluation)
```

non tail recursif

```
WHILE1: XCHG              ; HL ← le progn à faire
        CALL  PRØGN
        PØP   HL          ; restaure le tout.
WHILE:  push  HL          ; sauve le tout
        CALL  EVPRED      ; evalue le CAR
        JNZ   WHILE1      ; ça boucle
        PØP   DE          ; recup le tout
        RET               ; voilà
```

CØNS (boxed) ZSUBR XCØNS NCØNS


```
NCØNS:                      ; (CØNS HL NIL) → HL
[167]  10   LXI   DE,0      ; NIL → DE
            JMP   CØNS1     (struck through)

CØNS:                       ; (CØNS HL DE) → HL
[157]  4    XCHG

XCØNS:                      ; (CØNS DE HL) → HL
[153]  5    MØV   B,H   }
       5    MØV   C,L   }   cdr → BC

CØNS1:
       16   LHLD  FREE      ; recup  Free □→ □□→ □□→ - - - □□  liste libre
       5    MØV   A,H   }
       4    ØRA   L     }   test fin
       10   JZ    GC    }   liste libre
       11   PUSH  HL        ; sauve le resultat du cons.
       7    MØV   M,E   }
       5    INX   HL    }
       7    MØV   M,D   }   change le CAR
       5    INX   HL    }
       7    MØV   E,M   }
       7    MØV   M,C   }
       5    INX   HL    }   change le CDR recup new free. (DE ←)
       7    MØV   D,M   }
       7    MØV   M,B   }
       4    XCHG            ; HL ← new free.
       16   SHLD  FREE
       10   PØP   HL        ; recup le resultat du cons.
       10   RET             ; voilà.
```

33 octets

EVLIS 1SUBR

LIST FSUBR

```
EVLIS:
LIST:   MØV     A,H   }
        ØRA     L     }   [] = NIL
        RZ            }
9       UNCØNS  HL,HL,DE
EVARG   PUSH    DE      ; sauve le reste
        CALL    EVAL    ; evalue le 1er element.
        ~~[struck out]~~
        ~~[struck out]~~
        CALL    NCØNS   ; crée le 1er doublet.
        PØP     DE      ; recupère le reste
        MOV     A,D   }
        ØRA     E     }  retour si ya q'1 element
        RZ            }
        Push    HL      ; sauve le 1er doublet
        XCHG            ; HL ← reste des element ; DE point. cour.
        Push    DE      ; sauve point courant
LIST1:
9       UNCØNS  HL HL DE
        Push    DE      ; sauve le reste   EVARG
        call    eval
        Call    NCØNS
        pop     BC      ; recup le reste
        Pop     DE      ; recup point cour
        XCHG            ; DE ← result du cons, HL ← point cour
        INX     HL    }
        INX     HL    }
        MØV     M,E   }  rplacd point.courant avec resultat.
        INX     HL    }
        MØV     M,D   }
        MØV     H,B   }  HL ← BC, le reste des elem
        MØV     L,C   }
        MØV     A,H   }
        ØRA     L     }  teste fin de liste
        JNZ     LIST1 }
        PØP     HL      ; recup le 1er doublet
        RET             ; voilà.
```

ya q'1 CØNS sya q'1 element

~~59 octects~~

39 EVARG

```
CDDDR:   5 INR  L
[112]    5 INR  L
CADDR:   7 MØV  A,M
[102]    5 INR  L
         7 MØV  H,M
         5 MØV  L,A
--------------------------
CDDR:    5 INR  L
[78]     5 INR  L
CADR:    7 MØV  A,M
[68]     5 INR  L
         7 MØV  H,M
         5 MØV  L,A
CDR:     5 INR  L
[44]     5 INR  L
CAR:
QUØTE:   7 MØV  A,M
[34]     5 INR  L
         7 MØV  H,M
         5 MØV  L,A
        10 RET
--------------------------
CDDAR:   5 INR  L
[112]    5 INR  L
CADAR:   7 MØV  A,M
[102]    5 INR  L
         7 MØV  H,M
         5 MØV  L,A
--------------------------
CDAR:    5 INR  L
[78]     5 INR  L
CAAR:    7 MØV  A,M
[68]     5 INR  L
         7 MØV  H,M
         5 MØV  L,A
        10 JMP  CAR
--------------------------
CDAAR:   5 INR  L
[112]    5 INR  L
CAAAR:   7 MØV  A,M
[102]    5 INR  L
         7 MØV  H,M
         5 MØV  L,A
        10 JMP  CAAR

CDADR:   5 INR  L
[112]    5 INR  L
CAADR:   7 MØV  A,M
[102]    5 INR  L
         7 MØV  H,M
         5 MØV  L,A
        10 JMP  CADR
```

[52] octets

MEMBER   2SUBR   recherche 2

ya pas de MEMQ

```
          XCHG              ; HL ← l , DE ← a
MEMBER:   MØV   A,D
          CPi   BNUM
          JC    MEMB5

          XCHG
MEMB1:    PUSH  HL          ; sauve L
          PUSH  DE          ; sauve a
          MØV   A,M
          iNR   L
          MØV   H,M         ; (CAR l) → hl
          MØV   L,A
          CALL  EQUAL
          PØP   DE
          XRA
          ØRA   H           ; equal a ramené T
          JNZ   MEMB2
          PØP   HL          ; recup L
          iNR   L
          iNR   L
          MØV   A,M
          iNR   L           ; cdr L → hl
          MØV   H,M
          MØV   L,A
          XRA
          ØRA   H           ; fin l ?
          JNZ   MEMB1       ; non
          RET               ; ouaip -
MEMB2:    PØP   HL          ; recup le debut de la liste
          RET               ; voilà.

MEMB3:    iNR   L
MEMB4:    iNR   L
          MØV   A,M         ; cdr HL → HL
          iNR   L
          MØV   H,M
          MØV   A,L
MEMB5:    XRA
          ØRA   H           ; si HL = NIL retour
          RZ
          MØV   A,M
          CPR   E
          JNZ   MEMB3       ; raté
          iNR   L
          MØV   A,M
          CPR   D
          JNZ   MEMB4       ; raté
          DCR   L           ; ØK : repositionne HL
          RET               ; voilà
```

long avec EQUAL

rapide avec EQ open

modif 1

SETQ FSUBR | SET RPLACA RPACB 2SUBR | RPACB 3SUBR

```
SETQ1:  MØV   H,B
        MØV   L,C
SETQ:   UNCØNS HL HL DE
        PUSH  DE
        UNCØNS HL HL DE
        PUSH  DE
        CALL  EVAL
        PØP   BC
        PØP   DE
        XCHG
        MØV   M,E
        INR   L
        MØV   M,D
        MØV   A,B
        ØRA   C
        JNZ   SETQ1
        XCHG
        RET
```

```
SET:
[33]   7  MØV   M,E
       5  INR   L
       7  MØV   M,D
       4  XCHG
      10  RET
```

```
RPLACA:
[34]   7  MØV   M,E
       5  INR   L
       7  MØV   M,D
       5  DCR   L
      10  RET
```

```
RPLACB:
[116]  4  XCHG       HL ← CDR  DE ← CAR
      18  XTHL       HL ← L
       7  MØV   M,E  }
       5  INR   L    }
       7  MØV   M,D  }  rplaca L CAR
       5  DCR   L    }
      10  PØP   DE
RPLACD: }
```

```
RPLACD:
[60]  11  PUSH  HL
       5  INR   L
       5  INR   L
       7  MØV   M,E
       5  INR   L
       7  MØV   M,D
      10  PØP   HL
      10  RET
```

modification 2

NEXTL FSUBR

NEXTL : (NEXTL at) (at) ← HL.

```
7  MØV  A,M   }
5  INX  HL    }  CAR HL → HL
7  MØV  H,M   }
5  MØV  L,A   }

7  MØV  E,M   }
5  INX  HL    }  Cval → DE
7  MØV  D,M   }

5  DCX  HL
4  XCHG          HL ← Cval ; DE ← @ de l'at.

7  MØV  C,M   }
5  INX  HL    }
7  MØV  B,M   }  Car Cval → BC que l'on sauve
5  INX  HL    }
11 PUSH BC    }

7  MØV  C,M   }
5  INX  HL    }  cdr Cval → BC
7  MØV  B,M   }

4  XCHG          HL ← l'@ de l'atom

7  MØV  M,C   }
5  INX  HL    }  rplaca at avec cdr Cval
7  MØV  M,B   }

10 POP  HL       recup car Cval
10 RET           voilà
```

23 octets

149

NULL ATØM LITATØM NUMBP LISTP	EQ NEQ = /=
1SUBR	2SUBR

```
NULL:    4 XRA     A
[38]     4 ØRA     H
        10 JZ      TRUE
FALSE:  10 LXI     HL, 0
        10 RET

NEQ:       LXI     BC,NULL
           PUSH    BC
EQ:        MØV     A,H
EQ1:       CPR     D          Eq de pointeurs
           JNZ     FALSE
           MØV     A,L
           CPR     E
           JNZ     FALSE
TRUE:      LXI     HL,'T
           RET

ATØM:    5 MØV     A,H
[42/52]  7 CPI     BLIST
     20+10 JNC     FALSE
     20+10 JMP     TRUE

LITATØM: 5 MØV     A,H
[42/52]  7 CPI     BNUM
     20+10 JNC     FALSE
     20+10 JMP     TRUE

NUMBP:   5 MØV     A,H
[42/49/59] 7 CPI   BNUM
     20+10 JC      FALSE
         7 CPI     BLIST
     20+10 JNC     FALSE
        10 RET

LISTP:     MØV     A,H
           CPI     BLIST
           JC      FALSE
           RET
```

```
NEQN:      LXI     BC, NULL
           PUSH    BC

EQN:       MØV     A,H
           CPI     BNUM
           JC      EQ1
           CPI     BLIST
           JNC     EQ1
           MØV     C,M        Eq de nombres
           INR     L
           MØV     B,M
           XCHG
           MØV     A,H
           CPI     BNUM
           JC      FALSE
           CPI     BLIST
           JNC     FALSE
           MØV     E,M
           INR     L
           MØV     D,M
           MØV     H,B
           MØV     L,C
           JMP     EQP
```

```
NEQUAL:   LXI    BC,NULL
          PUSH   BC

EQUAL:    PUSH   HL
          LXI    HL,0
          DADD   SP          ; sauve SP
          SHLD   SAVSP
          POP    HL
          JMP    EQUAL2

EQUAL1:   PUSH   DE
          XCHG
          LXI    HL,0
          Dadd   SP          ; test fin pile.
          MOV    A,H
          CPI    BSTACK
          JC     ERFS
          XCHG
          UNLONS HL DE HL
          XTHL
          PUSH   DE
          UNLONS HL DE HL
          XTHL
          CALL   EQUAL2
          XRA    A
          ORA    H
          JZ     EQUAL3
          POP    DE
          POP    HL
EQUAL2:   MOV    A,H
          CPI    BLIST       ; si HL ≠ LIST va EQ
          JC     EQN
          MOV    A,D
          CPI    BLIST       ; si DE = LIST va EQUAL1
          JNC    EQUAL1      ;    sinon faux
EQUAL3:   LHLD   SAVSP       ; retour rapide si faux
          SPHL
          JMP    FALSE
```

1+ 1- MINUS ABS	ADD SUB
1SUBR	2SUBR

```
1+:     7   mov   C,M                [164]
        5   inx   HL
        7   MØV   B,M
        5   inx   BC
        140 Jmp   cranum

1-:     mov   C,M
        inx   HL
        MØV   B,M
        DCX   BC
        Jmp   cranum

Minus:  mov   A,M
        inx   HL
MIN1:   CMA
        MØV   C,A
        MØV   A,M
        CMA
        MØV   B,A
        inx   BC
        Jmp   cranum

Abs:    mov   C,M
        inx   HL
        MØV   A,M
        ANI   /70
        RZ
        mov   A,C
        Jmp   min1
```

```
ADD:    mov   C,M
        inx   HL
        mov   B,M
        XCHG
        mov   E,M
        inx   HL
        mov   D,M
ADDS:   XCHG
        DADD  BC          [mov  B,H
        Jmp   cranum       mov  C,L]

SUB:    mov   C,M
        inx   HL
        MØV   B,M
        XCHG
        mov   A,M
        CMA
        mov   E,A
        inx   HL
        mov   A,M
        CMA
        mov   D,A
        inx   DE
        Jmp   ADDS
```

```
LENGTH: 7   LXI   BC,0
        10  Jmp   LNGT2
LNGT1:  5   inx   BC
        5   inx   HL
        5   inx   HL
        7   MØV   A,M
        5   inx   HL
        7   MØV   H,M
        5   MØV   L,A
LNGT2:  5   MØV   A,M
        4   ØRA   L
        [illegible]
        10  JNZ   LNGT1
        140 Jmp   cranum
```

157 + 58*n

f numériques à 2 args

1er arg → HL, 2ème arg → DE

```
NUM2:                     ; Met dans HL, DE les valeurs des 2 args
        MØV    A,M
        CPI    BATØM
        CC     CAR
        XCHG
        MØV    A,M
        CPI    BATØM
        CC     CAR
        ~~XCHG~~
        RET
```

```
PLUS:   CALL   NUM2     ~~XCHG~~
                        ~~XCHG~~
        DADD   DE
        JMP    CRANUM

DIFF:   CALL   NUM2
        MØV    A,D
        CMA
        MØV    D,A
        MØV    A,E
        CMA
        MØV    E,A
        INX    DE
        DADD   DE
        JMP    CRANUM

LØGSHIFT:
        CALL   NUM2
        MØV    A,E
        ANI    A/F
        RZ
LØGSH1: DADD   HL
        DCR    A
        JNZ    logsh1
        Jmp    cranum.
```

```
1+:     MØV   A,H
        CPI   BATØM
        CC    CAR
        INX   HL
        JMP   CRANUM

1-:     MØV   A,H
        CPI   BATØM
        CC    CAR
        DCX   HL
        JMP   CRANUM

+:      1er arg → HL, 2ème arg → DE

        MØV   A,H
        CPI   BATØM
        CC    CAR
++:     XCHG
        MØV   A,H
        CPI   BATØM
        CC    CAR
        DADD  DE
        JMP   CRANUM

-:
        MØV   A,H
        CPI   BATØM
        CC    CAR
        XCHG
        MØV   A,H
        CPI   BATØM
        CC    CAR
        MØV   A,H
        CMA
        MØV   H,A
        MØV   A,L
        CMA
        MØV   L,A
        INX   HL
        DADD  DE
        JMP   CRANUM

        XCHG
        MØV   A,H
        CPI   BATØM
        CC    CAR
        MØV   A,H
        CMA
        MØV   H,A
        MØV   A,L
        CMA
        MØV   L,A
        INX   HL
        JMP   ++
```

```
logshift:

        MØV   A,H
        CPI   BATØM
        CC    CAR
        XCHG
        MØV   A,H
        CPI   BATØM
        CC    CAR         XCHG
        ----------------------
        MØV   A,E
        ANI   A,/F
        RZ
    ┌─> DADD  HL
    │   DCR   A
    └── JNZ   ___
        JMP   CRANUM
```

Print & Prin1 FSUBR

pas de DMØ

mais des FØRMATS

(PRIN1 x (Format (TTAB 12)) Y)

→ XCHG

PRIN1 : UNCØNS HL HL DE
PUSH DE
si (CAR HL) = FØRMAT
alo PRØGN
cdr L
push HL
imprim
PØP HL
pop DE
si de ≠ NIL

~~(DC "!" (ξX)~~
~~(IF (EQ (Peekch) "!")~~

(DC "!" ()
['FØRMAT ξ (IF (EQ (Peekch) "!") ['TERPRI]
['TTAB (READ)]])

~~(PRINT "*" !5 X "*"~~

(Print "*" X !10 "*" !20 Y !30 "*")

REVERSE 2SUBR

```
REVERSE:    XCHG
            JMP     REV2

REV1:       XCHG
            MØV     E,M
            INX     HL
            MØV     D,M
            INX     HL
            MØV     C,M
            INX     HL
            MØV     B,M
```

CRANUM

CRANUM: cré un nb dans BC.

[130]

```
16  LHLD   FNUM
5   MØV    A,H
4   ØRA    L
10  JZ     GCN
11  Push   HL
7   MØV    M,B
5   iNR    L
7   MØV    M,C
5   iNR    L
7   MØV    A,M
5   iNR    L
7   MØV    H,M
5   MØV    L,A
16  SHLD   FNUM
10  PØP    HL
10  RET
```



EVARG

```
EVARG:  7 MØV   E,M
        5 INR   L
        7 MØV   D,M
        5 INR   L
        7 MØV   A,M
        5 INR   L
        7 MØV   H,M
        5 MØV   L,A
       18 XTHL
       11 PUSH  HL
        4 XCHG
       10 JMP   EVAL
```

```
52  UNCØNS  HL HL DE
11  PUSH    DE            13
17  CALL    EVAL
--
80

 17 CALL    EVARG          3
 91
 --
108
```

μ VLISP

f. stds.

I/O READ PRINT PRIN1 TERPRI MATCH

Interprète EVAL EPRØGN PRØGN EVLIS

Control IF CØND SELECTQ ØR AND WHILE
ESCAPE LESCAPE.

Predicat NULL ATØM NUMBP LISTP
EQ EQUAL

rech CAR CDR C..R ~~MEMQ~~ MEMBER ASSØC

P-L GET PUT [MAP MAPC MAPCAR]

créat CØNS LIST APPEND CØPY EXPLØDE READLIST

modif RPLACA RPLACD SETQ SET NCØNC NEXTL

Arith 1+ 1- + - * / \ LENGTH
> >= < <=

NIL T UNDEF λ Fλ μλ 1SUBR 2SUBR 3SUBR NSUBR FSUBR

DE DF DM DC

GETF SETF